Stanford Machine Learning - Week IX

Eric N Johnson

September 11, 2016

1 Anomaly Detection

1.1 Density Estimation

1.1.1 Problem Motivation

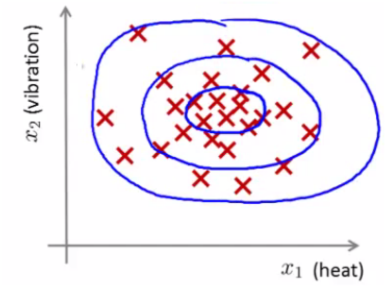
Here is an example of Anomaly Detection. Consider an aircraft engine manufacturing or maintenance operation:

$$\begin{aligned}\{x^{(1)}, x^{(2)}, ..., x^{(m)}\} &= \text{dataset}\\ &\text{where}\\ x_1 &= \text{heat generated}\\ x_2 &= \text{vibration intensity}\\ &\vdots\end{aligned}$$

When we receive a new set x_{test}, does it behave like the rest of the data or is it an anomaly?

Given a normal dataset, we will build a model $p(x)$ to describe it. Then we calculate probabilities that the data is anomalous:

$$\begin{aligned}p(x_{\text{test}}) < \varepsilon &\rightarrow \text{flag anomaly}\\ p(x_{\text{test}}) \geq \varepsilon &\rightarrow \text{Okay}\end{aligned}$$



This will help us determine a density estimation. **Anomaly Detection in Practice**

- Fraud detection

$$\begin{aligned}&x^{(i)} = \text{features of user } i\text{'s activities}\\ &\text{Model } p(x) \text{ from data}\\ &\text{Identify unusual users by checking which have } p(x) < \varepsilon\end{aligned}$$

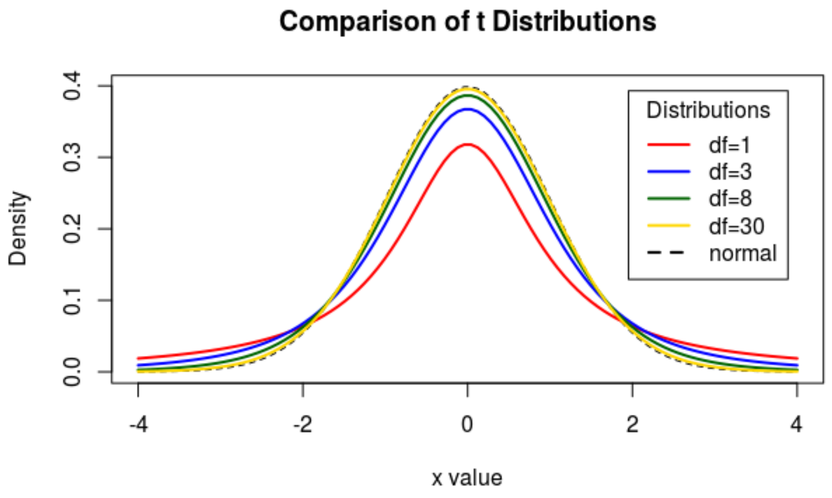
- Manufacturing (example above)
- Monitoring computers in a data center

$$\begin{aligned}&x^{(i)} = \text{features of machine } i\\ &x_1 = \text{memory use}\\ &x_2 = \text{CPU load}\\ &\vdots\\ &\text{Model } p(x) \text{ from data}\\ &\text{Identify unusual machines by checking which have } p(x) < \varepsilon\end{aligned}$$

1.1.2 Gaussian Distribution

Say $x \in \mathbb{R}$. If x is a distributed Gaussian with mean μ and variance σ^2. We say x is distributed:

$$x \sim \mathcal{N}(\mu, \sigma^2)$$



This is a plot of Gaussian Distributions with different degrees of freedom. The formula for a Gaussian distribution is:

$$p(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$
$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \mu\right)^2$$

the second formula may have an $m-1$ in the denominator (corrected sample standard deviation).

1.1.3 Algorithm

For our training set: $\{x^{(1)}, ..., x^{(m)}\}$ where each $x \in \mathbb{R}$, we model $p(x)$ based on an assumption of independence of each x_i:

$$\begin{aligned} p(x) &= p(x_1; \mu_1, \sigma_1^2)p(x_2; \mu_2, \sigma_2^2)\cdots p(x_m; \mu_m, \sigma_m^2) \\ &= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) \end{aligned}$$

We assume $x_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and calculate the mean and variance of each feature according to

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$
$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_j^{(i)} - \mu_j\right)^2$$

Anomaly Detection Algorithm:

1. Choose features x_i that you think might be indicative of anomalous examples. These will have unusually large or unusually small values.

2. Given the training set $\{x^{(1)}, ..., x^{(m)}\}$, fit parameters $\mu_1, ..., \mu_n, \sigma_1^2, ..., \sigma_n^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$
$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_j^{(i)} - \mu_j\right)^2$$

3. Given new example x, calculate $p(x)$:

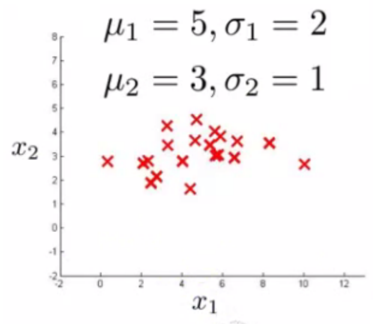
$$\begin{aligned} p(x) &= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) \\ &= \prod_{j=1}^{n} \frac{1}{\sigma_j\sqrt{2\pi}} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right) \end{aligned}$$

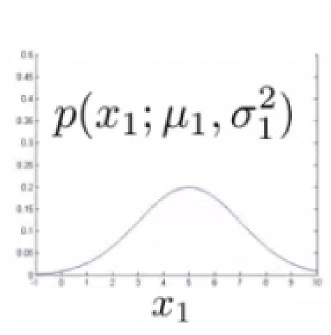
4. Flag anomaly if $(p(x) < \varepsilon$.

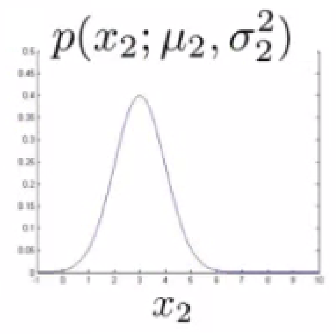
Note: A vectorized version of the mean in (2) will look like:

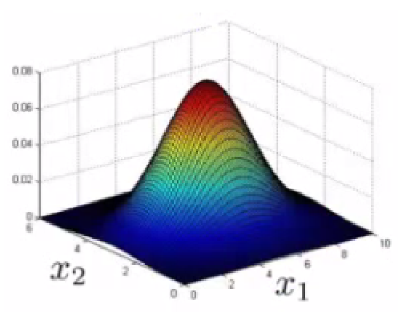
$$\begin{aligned} \mu &= \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} \\ &= \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \end{aligned}$$

Here are some pictures from an example:



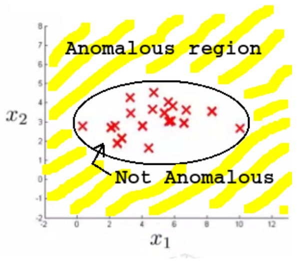






To figure out if $x_{\text{test}}^{(1)}$ and $x_{\text{test}}^{(2)}$ are anomalous, we set $\varepsilon = 0.02$, and calculate

$$p\left(x_{\text{test}}^{(1)}\right) = 0.0426 \geq \varepsilon \quad (\therefore \text{ not an anomaly})$$
$$p\left(x_{\text{test}}^{(2)}\right) = 0.0010 < \varepsilon \quad (\therefore \text{ an anomaly})$$



1.2 Building an Anomaly Detection System

1.2.1 Developing and Evaluating an Anomaly Detection System

It is very important to have some kind of real-number evaluation of algorithms. When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our system.

- Assume we have some labeled data containing both anomalous and non-anomalous examples ($y = 0$ if normal, $y = 1$ is anomalous.)

(i) Create a training set of non-anomalous examples:

$$x^{(1)}, x^{(2)}, ..., x^{(m)}$$

(ii) Create a cross-validation set:

$$\left(x_{cv}^{(1)}, y_{cv}^{(1)}\right), \left(x_{cv}^{(2)}, y_{cv}^{(2)}\right), ..., \left(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})}\right)$$

(iii) Create a test set:

$$\left(x_{test}^{(1)}, y_{test}^{(1)}\right), \left(x_{test}^{(2)}, y_{test}^{(2)}\right), ..., \left(x_{test}^{(m_{test})}, y_{test}^{(m_{test})}\right)$$

For example: Aircraft widgets manufactured by 'Aircraft Widget World, Inc.'

- We have 10,000 normal widgets ($y = 0$)
- We have 20 screwy widgets ($y = 1$)

Split this up as follows:

(i) Training set: 6000 good widgets. These are used to find $\mu_1, \mu_2, ..., \mu_n, \sigma_1^2, \sigma_2^2, ..., \sigma_n^2$ and thus calculate

$$p(x) = p(x_1; \mu_1, \sigma_1^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

(ii) Cross validation set: 2000 good widgets, and 10 defective widgets.

(iii) Test set: 2000 good widgets, 10 defective widgets.

Algorithm Evaluation

Fit model $p(x)$ on training set $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$

On cross validation / test example, predict

$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \quad \text{(Anomaly)} \\ 2 & \text{if } p(x) \geq \varepsilon \quad \text{(normal)} \end{cases}$$

Use an evaluation metric:

Truth table: $\{t_p, t_n, f_p, f_n\}$

Precision / recall

$F_1 -$ score

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \text{where}$$

$$\text{precision} = \frac{t_p}{t_p + f_p}$$

$$\text{recall} = \frac{t_p}{t_p + f_n}$$

We can also use the cross-validation set to choose the value of the parameter ε

1.2.2 Anomaly Detection vs. Supervised Learning

If we already have a set that contains known labels for a bunch of items, why don't we use supervised learning (like logistic regression) to test unknown items?

Anomaly Detection

+ Very small number of (positive $y = 1$) examples
+ Large number of normal (negative $y = 0$) examples
+ We have several different types of anomalies. This is because it is difficult for any algorithm to learn anything useful from anomalous examples. Future anomalies may look nothing like anything the algorithm has seen so far.

Applications:

- Fraud detection
- Manufacturing
- Monitoring machines in a data center

Supervised Learning

+ Large number of both positive and negative examples
+ Enough positive examples for algorithm to sense what positive examples look like...
+ Future anomalies will be similar to ones from the training set.

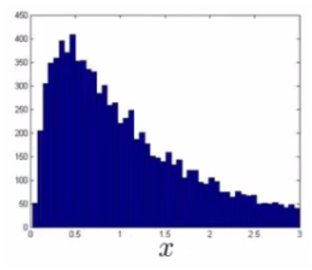
Applications:

- Email spam classification
- Weather prediction
- Medical classification

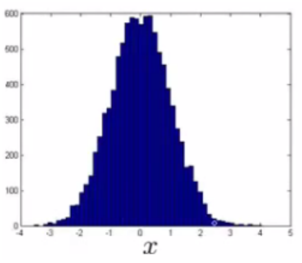
1.2.3 Choosing What Features to Use

One thing that has a huge effect on the effectiveness of an anomaly detection algorithm is which features we choose to give to our algorithm.

- Gaussian features are best. We can plot histograms to see whether the distributions are skewed. If they are, we may decide to transform data using logarithmic transformations. For example, a feature may have a histogram that looks like:



If we transform this with $x_1 \leftarrow \log(x_1)$, we get something more Gaussian.



I.e., we might try

$$\begin{aligned} x_1 &\leftarrow \log(x_1) \\ x_1 &\leftarrow \log(x_1 + c) \text{ for some } c \in \mathbb{R} \\ x_1 &\leftarrow \sqrt{x_1} \\ x_1 &\leftarrow x_1^4 \end{aligned}$$

- Perform an error analysis for anomaly detection.
  - We want $p(x)$ to be large for normal examples x.
  - We want $p(x)$ to be small for anomalous examples x.

  The most common problem is that $p(x)$ is comparable (large or small for both) for normal and anomalous examples.
- Choose features that will take unusually large or small values in the event of an anomaly. For example, if

$$\begin{aligned} x_1 &= \text{RAM use} \\ x_2 &= \text{CPU load} \\ x_3 &= \text{Network traffic} \end{aligned}$$

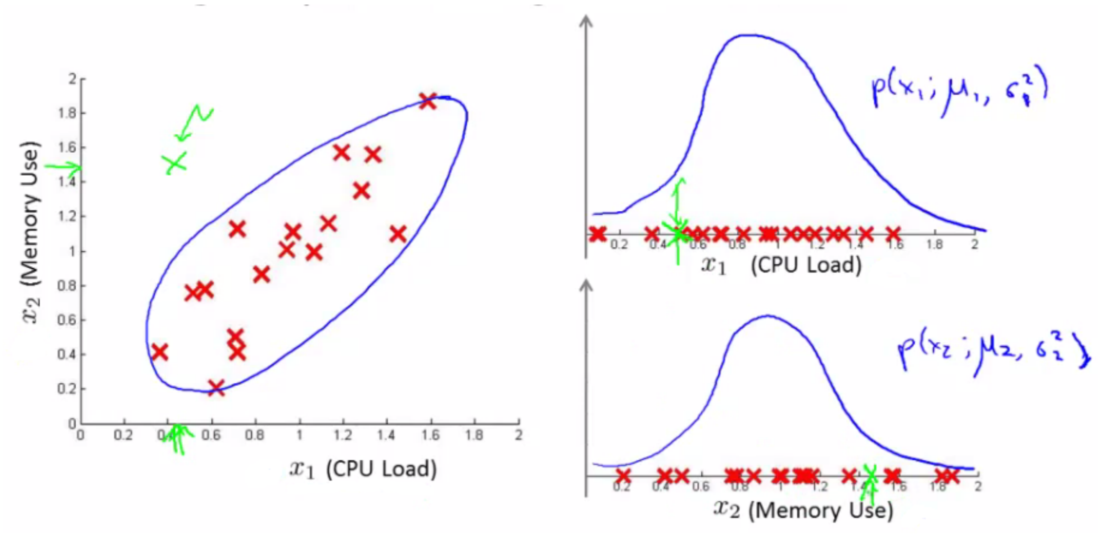
  we may find that

$$\begin{aligned} x_4 &= \frac{x_1}{x_3}, \text{ or} \\ x_5 &= \frac{x_1^2}{x_3} \end{aligned}$$

  have this property.

1.3 Multivariate Gaussian Distribution

Consider an example of machines in a data center.

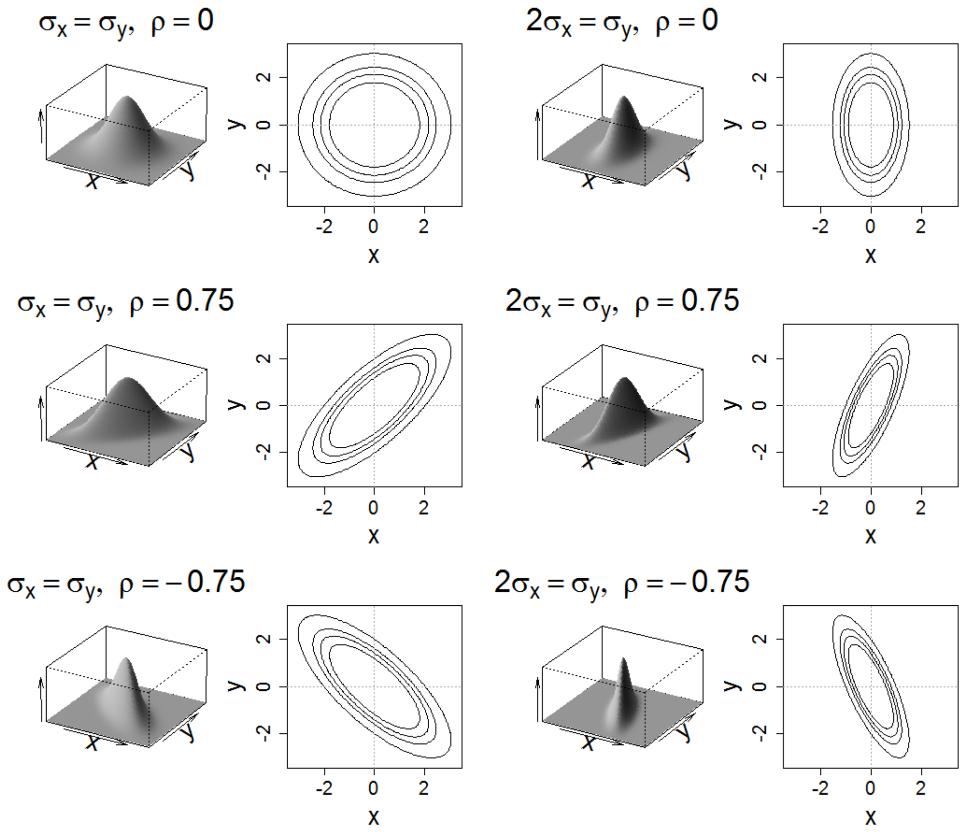


An anomaly detection algorithm may fail to detect this ‘green’ point because neither $p(x_1; \mu_1, \sigma_1^2)$ or $p(x_2; \mu_2, \sigma_2^2)$ is all that unlikely. However, the data point is fairly far from the others. To fix this we can use a multivariate Gaussian distribution.

Multivariate Gaussian Distribution

Given training set $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$, for $x \in \mathbb{R}^n$, and we model $p(x)$ all at once. Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (Covariance matrix).

$$\begin{aligned}
p(x; \mu, \Sigma) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right), \text{ where} \\
|\Sigma| &= \det(\Sigma), \\
\mu &= \frac{1}{m}\sum_{i=1}^{m} x^{(i)}, \\
\Sigma &= \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T
\end{aligned}$$



The vector $\mu = \begin{bmatrix} \mu_1 & \mu_2 & \dots & \mu_m \end{bmatrix}^T$ determines the center of the distributions. The quantity ρ is the correlation $\rho = cor(x_i, x_j)$. x_i and x_j where $i \neq j$ are independent if and only if $\rho = 0 \ \forall i, j$.

The variance-covariance matrix consists of the variances of the variables along the main diagonal and the covariances between each pair of variables in the other matrix positions. For a multivariate Gaussian of independent x_1, x_2, and x_3, if $\sigma_1 = 1$, $\sigma_2 = 1/2$, and $\sigma_3 = 2$, we get this covariance matrix $\Sigma \in \mathbb{R}^{3 \times 3}$:

$$\begin{aligned}
\Sigma &= \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 2 \end{bmatrix}
\end{aligned}$$

Note that Σ is symmetric positive definite. See "http://www.math.uah.edu/stat/special/MultiNormal.html"

1.3.1 Anomaly Detection using the Multivariate Gaussian Distribution

1. Fit model $p(x)$ by setting

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$
$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

2. Given a new example x, compute

$$p(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

3. Flag an anomaly if

$$p(x) < \varepsilon$$

We can prove that our multivariate Gaussian distribution is the same as:

$$p(x) = p(x_1;\mu_1,\sigma_1^2) \times p(x_2;\mu_2,\sigma_2^2) \times \cdots \times p(x_n;\mu_n,\sigma_2^2)$$

with some constraints on the shape of the distribution.

Comparison between models

Original Model

$$p(x) = p(x_1;\mu_1,\sigma_1^2) \times p(x_2;\mu_2,\sigma_2^2) \times \cdots \times p(x_n;\mu_n,\sigma_2^2)$$

- Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values.
- Computationally cheaper
- Okay even if the training set size m is small.

Multivariate Gaussian

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

- Automatically captures correlations between features
- Computationally more expensive
- Must have $m > n$ or Σ will be singular.

We may have a singular Σ in two cases:

- When $m > n$
- When features are linearly dependent.

We should fix either case before using the multivariate Gaussian.

---

2 Recommender Systems

2.1 Predicting Movie Ratings

If we let our users, for example, rate movies, can we suggest some new movies for them based on their ratings?

2.1.1 Problem Formulation

Let user movie ratings be in $\{0, 1, ..., 5\}$. And

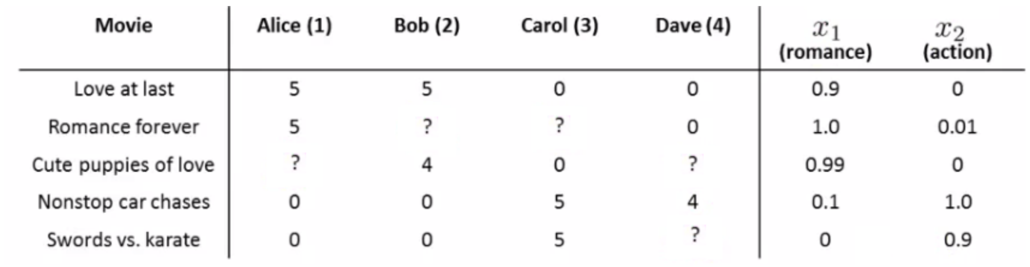
$$\begin{aligned} n_u &= \text{number of users} \\ n_m &= \text{number of movies} \\ r(i,j) &= 1 \text{ if user } j \text{ has rated movie } i \\ y^{(i,j)} &= \text{rating given by user } j \text{ to movie } i \text{ if } r(i,j) = 1 \end{aligned}$$

Given a users rating, can we recommend new movies to them?

2.1.2 Content-Based Recommendation

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

For example, suppose we have



Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

where we set features for each movie by setting $x_0 = 1$ (intercept), and specifying in advance like x_1 = 'amount of romance', and x_2 = 'amount of action'. So our feature vectors become

$$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}, x^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 0.01 \end{bmatrix}, x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}, \cdots$$

where $x_0 = 1$ is for our intercept. And suppose we know that the parameter vector for Alice contains

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}.$$

So to predict how Alice will rate the movie 'Cute Puppies of Love', we find

$$(\theta^{(1)})^T x^{(3)} = 5 * 0.99 = 4.95$$

How we find the parameter vector θ with some learning algorithm. Details to follow.

Problem Formulation

$$\begin{aligned}
r(i,j) &= 1 \text{ if user } j \text{ has rated movie } i\text{, zero otherwise}\\
y^{(i,j)} &= \text{rating by user } j \text{ on movie } i\text{, if defined}\\
\theta^{(j)} &= \text{parameter vector for user } j\\
x^{(j)} &= \text{feature vector for movie } i\\
m^{(j)} &= \text{number of movies rated by user } j
\end{aligned}$$

and for user j, movie i, the predicted rating is

$$(\theta^{(j)})^T(x^{(i)})$$

To learn the parameter vector $\theta^{(j)} \in \mathbb{R}^{n+1}$, we find

$$\min_{\theta^{(j)}} \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T(x^{(i)}) - y^{(i.j)}\right)^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} \left(\theta_k^{(j)}\right)^2$$

We will write this slightly differently and rewrite the optimization objective:

Optimization Objective:

- To learn $\theta^{(j)}$ (the parameter vector for user j), calculate:

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T(x^{(i)}) - y^{(i.j)}\right)^2 + \frac{\lambda}{2} \left(\theta_k^{(j)}\right)^2$$

- To learn $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$ (all at once), calculate:

$$\min_{\theta^{(1)},...,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T(x^{(i)}) - y^{(i.j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)}\right)^2$$

  which will give us a separate parameter vector for all users.

Using a **Gradient Descent Update**, each step is accomplished via:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} \qquad \text{for } k = 0$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \qquad \text{for } k \neq 0$$

The second term (after α) is the partial derivative of the optimization objective

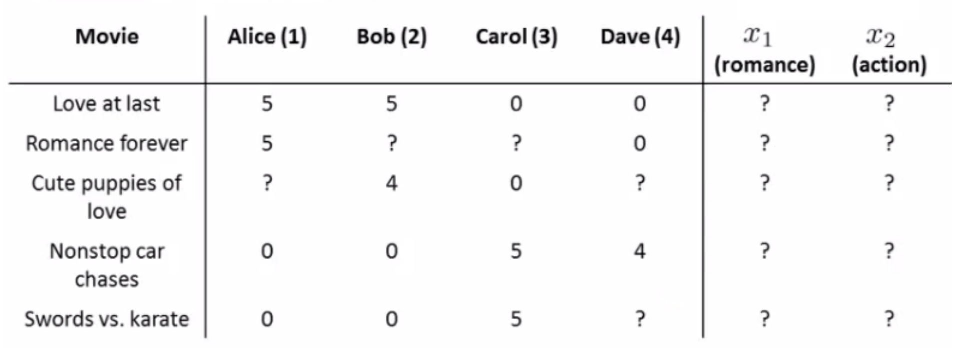
$$\frac{\partial}{\partial \theta_k^{(j)}} J\left(\theta^{(1)}, ..., \theta^{(n_u)}\right)$$

However, we may not have a way of filling in all of the details in the x vectors. For example, how 'action-oriented' or how 'romantic' it is. An algorithm that will learn these details is called **Collaborative Filtering**.

2.2 Collaborative Filtering

This is an algorithm that has the ability to learn about features. This will be useful if we can't characterize all of our movies.

Suppose we have a data set but we do not know in advance how 'action-oriented' or how 'romantic' a movie is.



Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

Suppose instead, the users supply some information about what kind of movies they like:

$$\theta^{(1)} = \begin{bmatrix} 0 & 5 & 0 \end{bmatrix}^T, \theta^{(2)} = \begin{bmatrix} 0 & 5 & 0 \end{bmatrix}^T, \theta^{(3)} = \begin{bmatrix} 0 & 0 & 5 \end{bmatrix}^T, \theta^{(4)} = \begin{bmatrix} 0 & 0 & 5 \end{bmatrix}^T$$

which correspond to the fact that Alice and Bob like romantic movies, but not action movies. Carol and Dave, however, like action movies but not romantic movies.

We can use these to infer the values of x_1 and x_2. Here is our optimization algorithm:

Optimization Algorithm

- Given $\theta^{(1)}, ..., \theta^{(n_u)}$, to learn $x^{(i)}$, calculate:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2$$

- Given $\theta^{(1)}, ..., \theta^{(n_u)}$, to learn $x^{(1)}, ..., x^{(n_m)}$ (all at once), calculate:

$$\min_{x^{(1)},...,x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2$$

In other words, given $r(i,j)$ and $y^{(i,j)}$ we can find the movie parameters with $x^{(1)}, ..., x^{(n_m)}$. Given these movie parameters, we can estimate $x^{(1)}, ..., x^{(n_m)}$. We may have to go back and forth between these to effectively get the results we need. However, there is a more efficient way to do this.

2.2.1 Collaborative Filtering Algorithm

Note that both of the 'inside' optimized functions are the same to find the theta parameters and movie features. The only difference is the order by which we perform the sums. So we will combine these and optimize both simultaneously.

Combined Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, ..., x^{(n_m)}$ and $\theta^{(1)}, ..., \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)})$ using Gradient Descent for every $j = 1, ..., n_n$ and $i = 1, ..., n_m$:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} \qquad \text{for } k = 0$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \qquad \text{for } k \neq 0$$

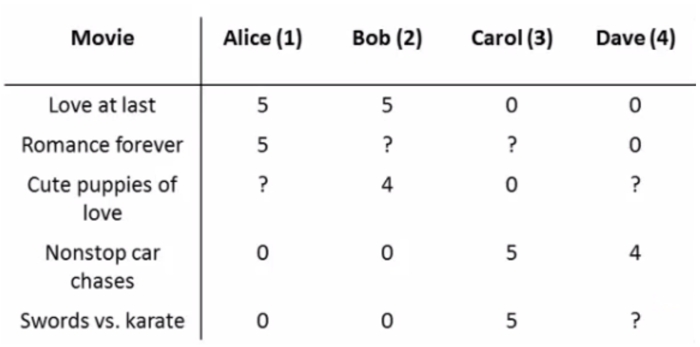
3. For a user with parameters θ and a movie with learned features x, predict a star rating for missing ratings with $\theta^T x$.

2.3 Low Rank Matrix Factorization

How do we vectorize and make this algorithm easy to run? There is a different way of writing out the predictions for the Collaborative Filtering Algorithm. This algorithm is called 'Low Rank Matrix Factorization.'

2.3.1 Vectorization: Low Rank Matrix Factorization

Here is our movie data again, which we will turn into a matrix.



Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & 4 \end{bmatrix}$$

Since $n_m = 5$ and $n_u = 4$, we have a 5×4 matrix. We still say that user j predicts of movie i using $(\theta^{(j)})^T x^{(i)}$. Thus, we have this matrix of predictive ratings:

$$\begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \cdots & (\theta^{(4)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \cdots & (\theta^{(4)})^T(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \cdots & (\theta^{(4)})^T(x^{(n_m)}) \end{bmatrix}$$

We may write this out in a simpler way. Let

$$X = \begin{bmatrix} --(x^{(1)})^T-- \\ --(x^{(2)})^T-- \\ \vdots \\ --(x^{(n_m)})^T-- \end{bmatrix}$$

and

$$\Theta = \begin{bmatrix} --(\theta^{(1)})^T-- \\ --(\theta^{(2)})^T-- \\ \vdots \\ --(\theta^{(4)})^T-- \end{bmatrix}.$$

And so we have

$$X\Theta^T = \begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \cdots & (\theta^{(4)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \cdots & (\theta^{(4)})^T(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \cdots & (\theta^{(4)})^T(x^{(n_m)}) \end{bmatrix}$$

This lends itself to a vectorized implementation to find all of these predictions.

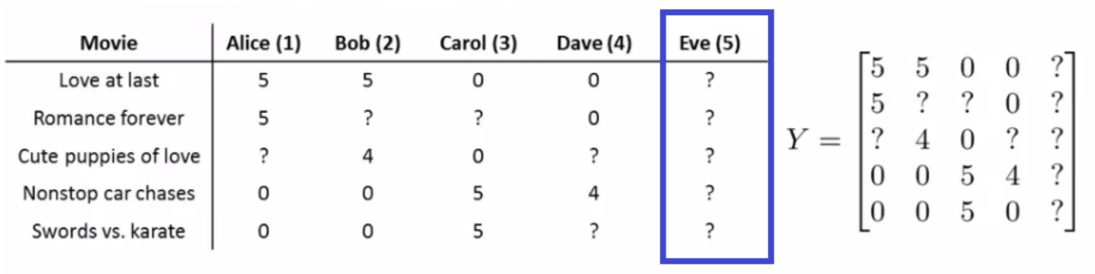
2.3.2 Finding related movies

For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$. These feature vectors may be something like 'x_1 = romance, x_2 = action, x_3 = comedy,' etc. How do we find movies j related to movie i? If some movie j is close (in some sense of distance) from movie i, it may be a good choice.

$$\text{small } |x^{(i)} - x^{(j)}|_2 \rightarrow \text{movie } i \text{ and } j \text{ are similar}$$

2.3.3 Implementational Detail: Mean Normalization

Mean Normalization will help the algorithm work better. What will happen if we have a new user who has not rated any movies yet?



Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
Love at last	5	5	0	0	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs. karate	0	0	5	?	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

Here is our summation:

$$\min \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

We will have to learn a new vector $\theta^{(5)} \in \mathbb{R}^2$ for our new user. The first term of the first summation is not used because $r(i,j)$ is never equal to 1. Also $\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ initially and so $(\theta^{(5)})^T x^{(i)} = 0$ everywhere. But having zeros for predictions for all movies is not very helpful. *Mean Normalization* will address this. We find the mean rating for all of the values found so far in Y.

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \rightarrow \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

Then for each rating in Y, we subtract off the mean rating in μ

$$Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

which gives all of the ratings a mean of zero. Then for user j on movie i, we predict

$$(\theta^{(j)})^T (x^{(i)}) + \mu_i$$

For Eve, this becomes the mean ratings given by everyone else:

$$\text{Eves Predicted Ratings} = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$